NASA Technical Memorandum 4487

An Expert System Executive for Automated Assembly of Large Space Truss Structures

Cheryl L. Allen
Langley Research Center
Hampton, Virginia

National Aeronautics and Space Administration

Office of Management

Scientific and Technical Information Program

1993

The use of trademarks or names of manufacturers in this report is for accurate reporting and does not constitute an official endorsement, either expressed or implied, of such products or manufacturers by the National Aeronautics and Space Administration.

Abstract

Langley Research Center developed a unique test bed for investigating the practical problems associated with the assembly of large space truss structures using robotic manipulators. The test bed is the result of an interdisciplinary effort that encompasses the full spectrum of assembly problems - from the design of mechanisms to the development of software. This paper describes the automated structures assembly test bed and its operation, details the expert system executive and its development, and discusses the planned system evolution. Emphasis is on the expert system implementation of the program executive. The executive program must direct and reliably perform complex assembly tasks with the flexibility to recover from realistic system errors. The employment of an expert system permits information that pertains to the operation of the system to be encapsulated concisely within a knowledge base. This consolidation substantially reduced code, increased flexibility, eased software upgrades, and realized a savings in software maintenance costs.

Introduction

Projected, crewed missions to the moon and Mars depart from previous space endeavors because the large vehicles involved require assembly and checkout in space. The construction of these vehicles requires extensive in-space operations that call for enhanced capabilities in assembly and servicing. To perform these functions with the limited crew resources available, a higher level of automation than that presently available must be realized. As a first step in this direction, Langley Research Center developed a unique test bed to investigate the practical problems associated with the automated assembly of large space truss structures using robotic manipulators. The research program is an interdisciplinary effort that encompasses the full spectrum of assembly problems - from the design of mechanisms compatible with automated operations to the definition of the software structures and algorithms required for their support.

This research program adheres to the following design requirements:

1. All system development, testing, and demonstration are performed using full-scale test hardware because full-scale testing is considered the only way to identify all the problems associated with automated assembly.
2. System design and automation are integrated and complementary technologies with solutions that are developed cooperatively.
3. The program is targeted toward a fully automated system with either an astronaut or earth-based operator as a monitor who is needed only when the robotic system encounters a problem that requires intervention or assistance. The last requirement describes a mode of operation known as supervised autonomy, which holds the most promise for the accomplishment of large construction tasks with the limited crew resources available on orbit.

The purpose of this paper is to describe briefly the automated structures assembly test bed and its operation, to detail the expert system executive and its implementation, and to discuss the system expansion under development. The emphasis of the paper is the application of expert system techniques to the program executive; however, the hardware components are described, and a narrative of the assembly process is presented as a basis for the description of the software and its functions.

Facility Description

The Automated Structures Assembly Laboratory (ASAL) is shown in figure 1. Figure 1(a) shows a schematic of the assembly system with the major components labeled, and figure 1(b) is a photograph of the facility in operation. The assembly system consists of a robot arm, a motion base system, two specialized end effectors, the assembly components for the structure, and storage canisters for those components. The ASAL uses commercially available equipment when possible to minimize cost and to ease modification as research needs dictate. The hardware system is a ground-based research tool designed to permit evaluation of assembly techniques, strut

and end effector components, computer software architecture and algorithms, and operator interface requirements.

The structure selected for assembly is a planar tetrahedral truss that supports hexagonal reflector panels. (See fig. 2.) The completed structure comprises a truss with 102 strut members, each 2 m long, and 12 panels approximately 2.3 m across the vertices. The structure was designed as a laboratory prototype to represent the structures that support functional surfaces on a number of planned or proposed missions, such as antennas and aerobrakes.

A brief description of the major components follows. The details of the facility hardware, performance characteristics, and assembly procedures can be found in references 1–3.

Robot Arm

The robot arm is an electronically driven, six-degree-of-freedom industrial manipulator selected for its reach envelope, payload capacity, positioning repeatability, and reliability. The robot arm computer is based on a Motorola 68000 microprocessor, and all robot motions are programmed in a modified BASIC programming language. No modifications have been made to the manipulator other than those available from the manufacturer.

Motion Base System

The motion base system includes a linear translational x-y Cartesian carriage and a rotating turntable. The robot arm is mounted on the carriage, and the structure is assembled on the rotating turntable. (See fig. 1.) Motion base drive motors on all three axes are commanded by an Intel 80286 microprocessor-based indexer.

End Effectors

The end effectors, shown in figure 3, are specialized tools mounted on the robot arm that perform the strut and panel installation and removal operations. Figure 3(a) shows the strut end effector, and figure 3(b) shows the panel end effector. All end effector operations are controlled by an onboard microprocessor mounted near the robot arm wrist. Typical microprocessor operations are detailed in reference 4. All end effector mechanisms are equipped with simple sensors such as microswitches and linear potentiometers to monitor end effector operations so that the operator can be notified if a problem occurs. The processor is programmed in ANSI-compatible C and includes sufficient inout/output (I/O) to monitor the sensors associated with the operations of the end effector mechanism. A commercial force/torque load cell is mounted between the end effector and the robot arm to provide compliant movement capability during strut retrieval and installation operations.

Truss and Panel Elements

The truss joint and the node design for the truss assembly are shown in figure 4. The joint is composed of two parts: the connector (consisting of a face and a plunger), which is bonded to the graphite-epoxy tube to form a strut, and the receptacle, which is mechanically attached to the node. The strut end effector uses pneumatically actuated receptacle fingers to grasp passive guidance v-grooves on the receptacles, providing stability during strut installation and removal (ref. 5). After the end effector inserts the connector into the node receptacle, locking nuts are turned by a small electric gear-head motor to secure the strut into place. Assembly begins by connecting struts to the three nodes that are premounted on the motion base turntable.

As the truss assembly progresses, the panels are placed on nodes at the top of the truss using the panel end effector (ref. 6). The panel is an aluminum hexagonal frame with a reflective covering. The panels are positoned, then locked into place using end effector actuator pins.

Storage Canisters

The struts are stored in nine trays that are stacked in the working canister directly behind the robot arm. Each tray is fitted with handles that allow the strut end effector to remove empty trays from the working canister and transfer them to the storage canister located at one side of the robot arm.

The panels are stored vertically in a large canister at one end of the y carriage. The same actuator pins that are used to attach the panels to the truss are also used to secure the panels in the canister.

Assembly Procedure

The assembly process begins when the strut end effector acquires the first strut from the top tray in the working canister. After acquired, the strut is carried above the working canister, and the motion bases are positioned so that the robot arm can reach the required installation position. The robot arm then moves through a sequence of predetermined points to arrive at an approach point approximately

12 in. from the intended installation point in the structure. At this approach point, control is turned over to a machine vision system.

The machine vision system uses two small video cameras mounted on each end of the end effector to view targets placed on the receptacles, as shown in figure 4. The video image of the target is processed to distinguish the target from the background and to determine its position with respect to the camera. This information is used to direct robot arm moves toward the target location for strut installation. Details of the vision system can be found in reference 7. After the arm reaches the installation point, the vision system relinquishes control. Next, the end effector grapples the node receptacles in the structure, repositions the robot arm to reduce forces and torques at the end effector that are caused by minor positioning errors, and inserts and locks the strut into place. The robot arm then returns to the working canister for another strut.

After a specified number of struts have been installed, panels can be secured to the top of the structure. This task involves stowing the strut end effector by latching it to the tray in the top of the storage canister and picking up the panel end effector stored at one end of the panel canister. This end effector change is accomplished by a commercially available pneumatic, quick-change mechanism. Panels are retrieved using y-carriage motion base moves and are installed at predetermined points on top of the truss. Machine vision is not used for the placement of panels at this time.

Combinations of strut- and panel-installation sequences are executed until the structure is completed with 102 struts and 12 panels.

System Control and Communications

The ASAL facility is managed by several digital computers that are serially connected through RS232 communication lines. The program executive and operator interface functions are performed on a minicomputer. The robot arm motions, carriage movements, and end effector operations, as well as the computations required by the vision system, are executed on individual processors.

Software Design

The design layout for the assembly system software is illustrated in figure 5 and detailed in reference 8. The software is arranged in four hierarchical levels of commands (administrative, assembly, device, and component). Each level decomposes into a sequence of commands for the next lower level. The preliminary setup of the system is performed at the highest, or administrative, level. The operator can examine and modify data and system options and can select, create, and modify command and assembly sequence files. A goal-directed task sequence planner is intended to interface with this level. Currently the assembly sequence is manually determined and maintained in a file. Each entry in the assembly sequence file represents an appropriate assembly-level command (see fig. 5), which specifies the operations to be performed on a given element (that is, strut or panel). The standard operating mode is centered at the assembly level and reflects the automated aspects of the system. At this level the software manages all the devices, data verification, and error recovery. The assembly-level commands decompose into a series of commands for each of the three devices: the motion bases, the robot arm, and the end effector.

Although the assembly system is intended to operate in a fully automated mode, it is imperative that the operator is provided with sufficient internal information and has command access and authority at all levels to deal effectively with assembly errors. The operator completely controls error recovery and makes the final decision on error resolution. The operator can decide that an error is not severe, then command the system to proceed. Also, if none of the recovery options presented are successful, the operator can instruct the system to abort the failed operation and automatically roll the assembly process back to a known, successful condition. During assembly operations, the operator can pause the assembly process at any point and examine system details using a video display before either continuing or reversing the sequence.

The executive portion of the assembly system software directs and monitors assembly-level operations across the different processors and reports current status information to the operator. The executive maintains the conditions and constraints of the assembly operations such as details of the geometry of both the structure and the storage canisters. During an assembly, the executive determines what end effector to use and maintains the procedures required for its use. Finally, the executive tracks possible problems and recovery techniques for all assembly scenarios. To perform these functions effectively, the executive has full access to the current status of the assembly operation and the system hardware, which includes complete, detailed descriptions of the state of the assembled structure, the motion base, the robot arm, and the end effector hardware. This

information is continuously updated, based on sensor verification.

Initially, the assembly system software was written in FORTRAN. The procedural language was familiar to developers in ASAL so could be used to verify and refine assembly system operations relatively quickly. The initial task was to construct a simplified structure of 102 struts, using a single, premounted end effector. The functions of the end effector were commanded by the robot arm computer. The robot arm moved to predefined installation positions without the machine vision system. However, as the scope of the research project grew when panels, a second end effector, and distributed processors were added, the complexity of the information to be managed by the assembly system software increased. Because traditional programming languages were slow to keep up with system upgrades, portions of the software were rewritten using an expert system. The first level of code targeted for this transition was the decision-intensive program executive. The following sections describe the application of expert system techniques to the executive. Examples are presented.

Expert System Executive

The task of the program executive is to decide what actions to take (and the order in which to take them) during the construction of a given structure. To make informed decisions, the executive must have access to all current system information and the ability to evaluate that information in light of the desired task. This decision-making component of the assembly system software is best suited to implementation using expert system techniques.

Methodology

An expert system is a computer program that uses knowledge and reasoning techniques to solve problems that normally require the services of a human expert. A subset of the general area of expert systems concentrates on explicit representation of the knowledge of an expert about a class of problems, then provides a separate reasoning mechanism (called an inference engine) that operates on this knowledge to produce a solution. These systems are known as knowledge-based expert systems. The knowledge base is a file that contains the facts that compose expert knowledge about a specific domain. An inference engine is a program that applies reasoning techniques to the facts, as defined by the knowledge base, to draw conclusions. Inference engines vary according to the representation of the knowledge and the strategy for applying the knowledge.

A variety of expert system development tools are available to assist programmers in building powerful systems that can solve a wide range of problems. The commercially available Knowledge Engineering System (KES), as described in references 9 and 10, was selected for use in ASAL. The KES tool provides the inference engine, the knowledge representation schemes, and the facilities for creating an operator interface. The KES also provides an embedding utility to integrate expert systems with existing software by allowing the procedural language code to send, receive, and modify data from a knowledge base through special data types and run-time functions.

The KES inference engine uses rules to represent knowledge. This knowledge representation scheme is particularly well-suited to an application such as automated assembly, which organizes facts in the form of branching logic or if-then constructs. The KES uses deductive reasoning as the technique for problem solving; that is, certain outcomes follow directly from certain inputs.

The pursuit of a solution (or goal) drives the reasoning methodology used by the KES. This goal-driven, inferencing technique is known as backward chaining. Implicit subgoals are set up to determine values for attributes that appear in the antecedent of a rule that infers a value for some other attribute until a value for the goal attribute has been determined. In addition to goal-driven inferencing, the KES also performs event-driven inferencing through the use of demons. Event-driven (forward-chaining) inferencing occurs when the expert system responds to an event rather than pursues a goal.

The following section describes how this methodology was applied to the assembly system software in ASAL using the KES.

Implementation

As mentioned, the executive portion of the assembly system software was the first to be implemented as an expert system. The executive is responsible for managing all the devices (the motion bases, the robot arm, the end effectors, and the vision system), performing data verification, and enacting error recovery. Figure 6 illustrates where the knowledge base fits into the overall software system architecture. By embedding the knowledge base in the existing assembly system software, the executive has access to expert system methodologies for making decisions; at the same time, the familiar operator interface and existing data base management schemes are left intact.

The operator gains access to the executive through a menu-driven interface. By implementing a

menu-driven interface, the operator is presented with only the commands needed at a given time. As shown in figure 6, a layer of procedural code (FORTRAN and C routines) surrounds the knowledge base and handles the menu functions and information exchange between the knowledge base and the hardware. Data base information is also transferred through this surrounding code. The knowledge base contains the data constructs (attributes and classes), rules, and demons necessary to make informed decisions about the assembly process.

The expert system uses the knowledge base as the primary source to determine the command sent to a particular device at a given time. Commands associated with the specific hardware device are sent to the individual processors for interpretation and execution. Sensors are polled through device interfaces, and information is returned to the knowledge base when hardware status is needed. After a device-specific processor has completed a command, a return code is forwarded to the knowledge base so the next action can be sent. If the code is returned successfully, the data base is updated, and the next command in the sequence of assembly actions is determined. If an error occurs, instructions to return to the last known successful state may be issued. Information about all system functions is constantly updated and reported to the operator by way of status windows.

The structure and content of the knowledge base lies at the heart of the expert system; therefore, further consideration is warranted. The next sections detail the more important components of the knowledge base and present examples of their application.

Classes. The KES tool uses a structure called a class to describe a group of objects with the same set of characteristics. Each object is referred to as a member of the class, and each characteristic is maintained in a class construct known as an attribute. Two classes are defined in the current automated assembly knowledge base: one for struts and one for panels.

The strut class contains 102 unique members, one for each strut in the structure. The format of the class definition for struts, which includes the attribute declarations for strut members, is shown in figure 7. The attribute values associated with the physical aspects of the strut for the individual members are stored in a data base. As indicated in figure 7, 13 attributes are identified for struts: three that are associated with naming conventions (`OBSERVER NAME`, `ALTERNATE NAME`, and `ROBOT NAME`); two that identify the canister storage location (`TRAY`, `SLOT`); five that contain information about the physical characteristics of the strut (`NODE END`) and any special conditions for installation (`CAP END`, `FLIP`, `CAN_FLIP`, and `NODE DIRECT`); one that tracks the current location of the strut (`WHERE`); and two that define carriage positions of the robot arm during installation (`MB_INDEX1` and `MB_INDEX2`). Additional information about these attributes can be found in reference 8.

A class has also been defined for panels; this class contains information that pertains to the installed location for the panel and the panel installation status.

Rules. Rules form the knowledge source available to the inference engine. They represent expert knowledge, and they direct the actions of the expert system toward a desired goal. The general format of the rules is

`if` antecedent `then` consequent `endif`.

When the logical comparison in the antecedent is true, the rule "fires," and the KES commands in the consequent are performed, which drives the system toward a goal. For the expert system executive, the rules formulated require knowledge of the physical operations, the potential system states, and the capabilities of the hardware. Rules have been defined to capture information that pertains to tray-transfer operations and path-segment selection for strut- and panel-installation and/or removal operations.

The path the robot arm travels, from a rest position above the working canister to the installation point in the structure, is defined by a number of states. Figure 8 presents two rules that are used to determine the next state (`next_state`) in the installation path for a strut. For this illustration, the robot arm is poised above the working canister, waiting for directions to proceed to the grasp point of the current strut (`current_strut`) in the canister. The current location of the robot arm (`current_state`) and the direction of the robot arm motion (phase) determine the next state in the path of the robot. The robot phase (either into or out of the structure) is determined from the current location of the robot arm (`current_state`), the current location of the strut (`current_strut>where`), and the task or goal specified by the operator (`target_state`). The current location of the robot arm is maintained in a data base, and the location of the strut is held within the class member for that strut. To determine whether the consequence of the `state` rule is performed, the

phase rule must be evaluated. The firing of a rule often depends on the satisfaction of other rules. This backward-chaining technique makes rules extremely powerful.

The strut-installation path from the grasp point of the strut at the canister through the installation point at the structure and back requires 22 rules. The total knowledge base currently contains 59 rules: 22 rules to determine strut assembly paths previously indicated, 22 for panel paths, and 15 to transfer trays to and from the working canister to the storage canister.

Demons. Demons provide a method for event-driven inferencing within KES. The rules actively seek additional information to satisfy a specific goal, whereas demons remain passive until an event occurs that initiates execution. Demons modularize the procedural portions of the knowledge base and are useful for monitoring attributes for new or changed values.

A demon is composed of two parts: a guard and a body. A guard is similar to the antecedent of a rule and contains conditional statements to be evaluated. The body contains a list of commands that KES executes sequentially. Assigment of a new value to an attribute in a guard constitutes an event, which causes all associated demons (that is, demons with that attribute in their guard) to be evaluated. If the guard is true, then KES executes the commands in the body of the demon. In the expert system executive, when a value is assigned to an attribute in the consequent of certain rules, a demon is activated, which initiates event-driven inferencing.

For example, suppose the `state` rule of figure 8 is true, and the next state in the strut installation path is determined to be the canister grasp point (`GP_CAN`). The assignment of `GP_CAN` to the `next_state` attribute causes the demon in figure 9 to be evaluated. This demon is used to generate the command strings necessary to move the robot arm to `GP_CAN`. After some preliminary flags are set, the demon executes as follows (see fig. 9):

(a) By assigning a value of true to the attribute `check_scar`, another demon is activated, which ensures that the end effector is in the configuration necessary to make a safe approach to the canister.

(b) The value returned by the end effector is stored in the attribute `ee response`, which is examined before continuing.

(c) An uncorrectable error during the end effector operation would cause a rollback of the system to the last successful state.

(d) A successful return from the end effector permits the expert system to send a command to the processor associated with the robot arm to reset the force/torque load cell.

(e), (f), and (g) Installation conditions for the current strut are examined, and the command string is synthesized.

(h) and (i) The slot and tray numbers for the current strut are appended to the command string, and the command is sent to the robot. The assignment of true to the `send_merlin` attribute constitutes an event which activates yet another demon, which sends the command and evaluates the robot response.

(j) If the device operates successfully, the current state is updated. The `message` command is the means for sending the new value for the robot arm state to the data base through the embedded interface.

(k) and (l) An unsuccessful robot operation results in a reverse, or rollback, to the last known successful state.

A demon can change the value of the attribute that triggered its execution and resulted in recursive behavior. The body of a demon can also determine the value of another attribute that may itself contain associated demons. (See items (a) and (i) in fig. 9.) These demons can be triggered, which invokes forward chaining. By blending forward and backward chaining in a recursive environment, the expert system executive has evolved into a concise and powerful mechanism for representation of assembly knowledge.

Benefits

The concise representation afforded by the rule-based system reduced the lines of code significantly compared with the procedural (FORTRAN) version. In the initial implementation, the strut-installation path from the approach point above the working canister to the installation point in the structure and back required 19 rules and 26 demons. Each rule requires 3 lines of code; with the demons, about 445 executable lines of code were required for the implementation. These simplified constructs replaced approximately 1615 lines of executable FORTRAN, for a 72-percent reduction in code. Since this initial implementation, several additional capabilities have been added to the system (tray and panel operations, end effector changes, and machine vision), and the number of lines of code is still below that of the original FORTRAN version. This reduction in code increased maintainability and allowed rapid performance of modifications and upgrades.

A test was performed to assess the impact of adding an additional state in the path for a strut installation on both the FORTRAN version and the expert system version of the assembly system software. This modification is typical of the changes made to the software on a regular basis. In the FORTRAN version, 114 executable lines of code were added, and 22 existing lines were modified. These modifications affected five existing subroutines and required five additional subroutines. In the expert system version, 45 lines of code were encapsulated in 2 rules (3 lines each) and 2 demons for the addition of a new state. Five existing lines were also modified. The knowledge base was easier to debug and modify because the knowledge is separate from the algorithms and is readily accessible at run time, but the FORTRAN version was spread across a range of routines and required continued compilation. The FORTRAN programmer estimated approximately 8 man-hr to implement, debug, and test the change, yet the task was completed in the expert system in only 2 hr.

This structural assembly project is relatively simple compared with many of the in-space checkout and servicing tasks currently proposed. This first application of expert system techniques to the operations in ASAL has proven mandatory for effective system management. Knowledge-based methodologies are a requirement for the timely development and maintenance of these complex systems.

Research Opportunities

The goal of the ASAL research is to develop a complete integrated assembly system that incorporates on-line, automated planning and scheduling functions. The expert system executive described in this paper represents a first step in an evolution toward these advanced capabilities.

The expert system executive has successfully demonstrated the complete assembly of the 102-member truss structure with the 12 attached panels using machine vision and the microprocessor-controlled end effectors. This test verified the capabilities of the hardware and the software and established the utility of a supervised-autonomy operation mode. In addition, performance data were gathered that help direct the evolution of the system. Quantification of error recovery actions taken by the operator with the goal of automating many error recovery procedures is now in the work.

Currently, when an error occurs, a menu of potential solutions is presented to the operator. The operator must first assess the error by visually verifying sensor data, then select one or more options from an error recovery menu. By recording and studying such information as the operator choices, the state of the system when the error occurred, the order in which error recovery actions are attempted, and the successful actions as well as the failures, many processes can be automated. The final decision on error resolution still rests with the operator; however, some historically successful error recovery actions can be attempted before operator intervention is requested.

The enhancement with the largest software impact within ASAL is the changeover from the current system architecture to the highly distributed architecture depicted in figure 10. With this new architecture, the devices have their own processors and are controlled by an expert system scheduler. Maintenance of separate devices for individual processors allows for concurrent activity among many assembly operations.

Several advanced planners, each one with its own knowledge base, are also included in the design of the new architecture. In this system, knowledge bases exist for the following:

1. A task planner to develop assembly scenarios, based on a definition of truss geometry and stiffness characteristics
2. A tray-storage planner to determine strut storage and retrieval operations
3. A path planner to specify a collision-free path to the structure without reliance on predetermined approach points
4. A sequence planner to combine the information from the other planners to produce a script of assembly operations that highlights the concurrent actions

To manage the increased number of knowledge bases and individual processors, an advanced applications development tool known as the Strategic Networked Applications Platform (SNAP) was purchased from the supplier of KES. The SNAP tool supports the development of applications that operate in a distributed hardware environment. Existing KES applications can be directly converted into SNAP-compatible applications.

Concluding Remarks

The research conducted in the Automated Structures Assembly Laboratory (ASAL) successfully demonstrated the viability of using robotic manipulators to automatically assemble and disassemble large truss structures. During the construction of a given structure, the executive portion of the assembly system software is responsible for deciding

what actions to take and the order in which to take them. Due to the complexity of the executive software, continued implementation in traditional programming languages (i.e., FORTRAN) became prohibitive. Thus, preliminary investigation into the application of expert system technologies to perform the decision-making portions of the assembly software was extremely encouraging.

Future enhancements include implementation of a distributed architecture and several advanced planners. Multiple devices, each one with its own processors, will be controlled by an expert system scheduler. The addition of advanced planners with individual knowledge bases will establish an assembly system that promises to be both robust and reliable.

NASA Langley Research Center
Hampton, VA 23681-0001
June 22, 1993

References

1. Rhodes, Marvin D.; Will, Ralph W.; and Wise, Marion A.: *A Telerobotic System for Automated Assembly of Large Space Structures*. NASA TM-101518, 1989.

2. Rhodes, Marvin D.; and Will, Ralph W.: Automated Assembly of Large Space Structures. IAF Paper No. 90-272, Oct. 1990.

3. Will, Ralph; Rhodes, Marvin; Doggett, William R.; Herstrom, Catherine L.; Grantham, Carolyn; Allen, Cheryl L.; Sydow, P. Daniel; Cooper, Eric G.; Quach, Cuang C.; and Wise, Marion A.: An Automated Assembly System for Large Space Structures. *Intelligent Robotic Systems for Space Exploration*, Alan A. Desrochers, ed., Kluwer Academic Publ., c.1992, pp. 39–110.

4. Doggett, William R.; Rhodes, Marvin D.; Wise, Marion A.; and Armistead, Maurice F.: A Smart End-Effector for Assembly of Space Truss Structures. Paper presented at SOAR 91—Fifth Annual Space Operations, Applications, and Research Symposium, Final Program (Houston, Texas), July 9–11, 1991.

5. Wu, K. Chauncey; Adams, Richard R.; and Rhodes, Marvin D.: *Analytical and Photogrammetric Characterization of a Planar Tetrahedral Truss*. NASA TM-4231, 1990.

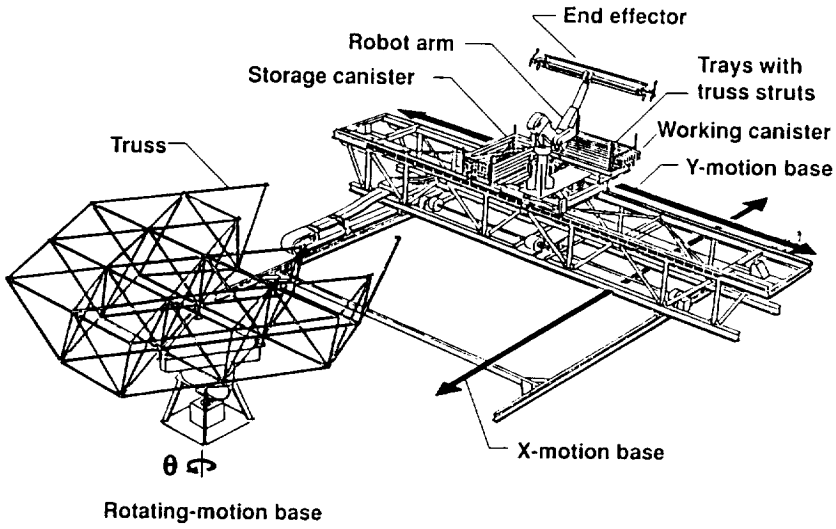
6. Kenner, W. Scott; Rhodes, Marvin D.; and Fichter, W. B.: *Component Count and Preliminary Assembly Considerations for Large Space Structures*. NASA TM-102604, 1990.

7. Sydow, P. Daniel; and Cooper, Eric G.: *Development of a Machine Vision System for Automated Structural Assembly*. NASA TM-4366, 1992.

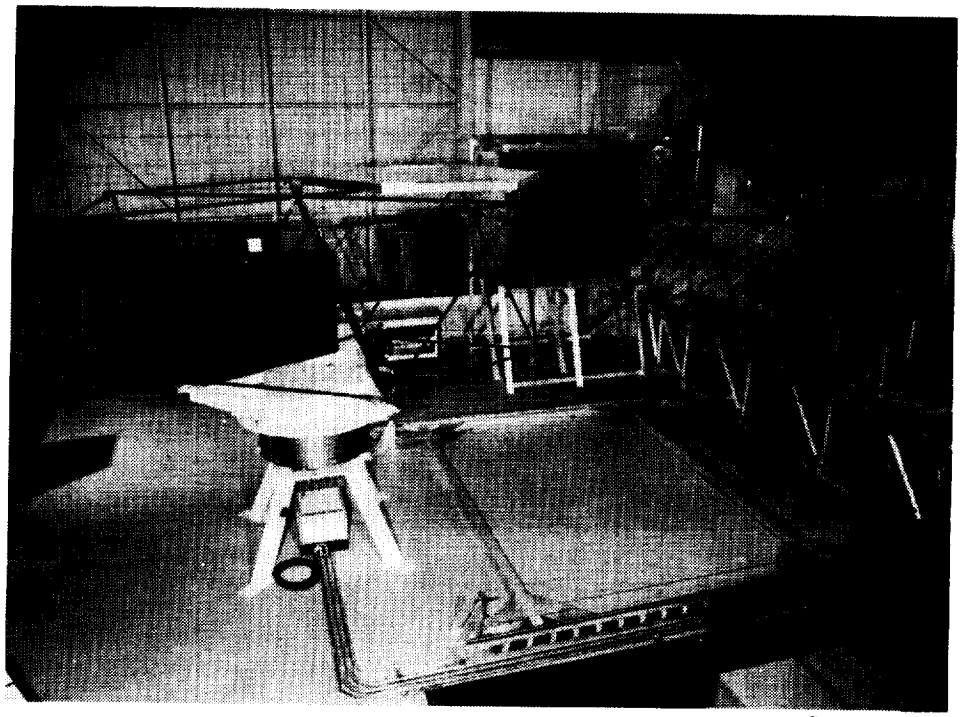
8. Herstrom, Catherine L.; Grantham, Carolyn; Allen, Cheryl L.; Doggett, William R.; and Will, Ralph W.: *Software Design for Automated Assembly of Truss Structures*. NASA TP-3198, 1992.

9. *Knowledge Base Author's Manual—KES PS*. Software Architecture & Engineering, Inc., c.1990.

10. *Knowledge Base Author's Reference Manual—KES PS*. Software Architecture & Engineering, Inc., c.1990.



(a) Schematic.



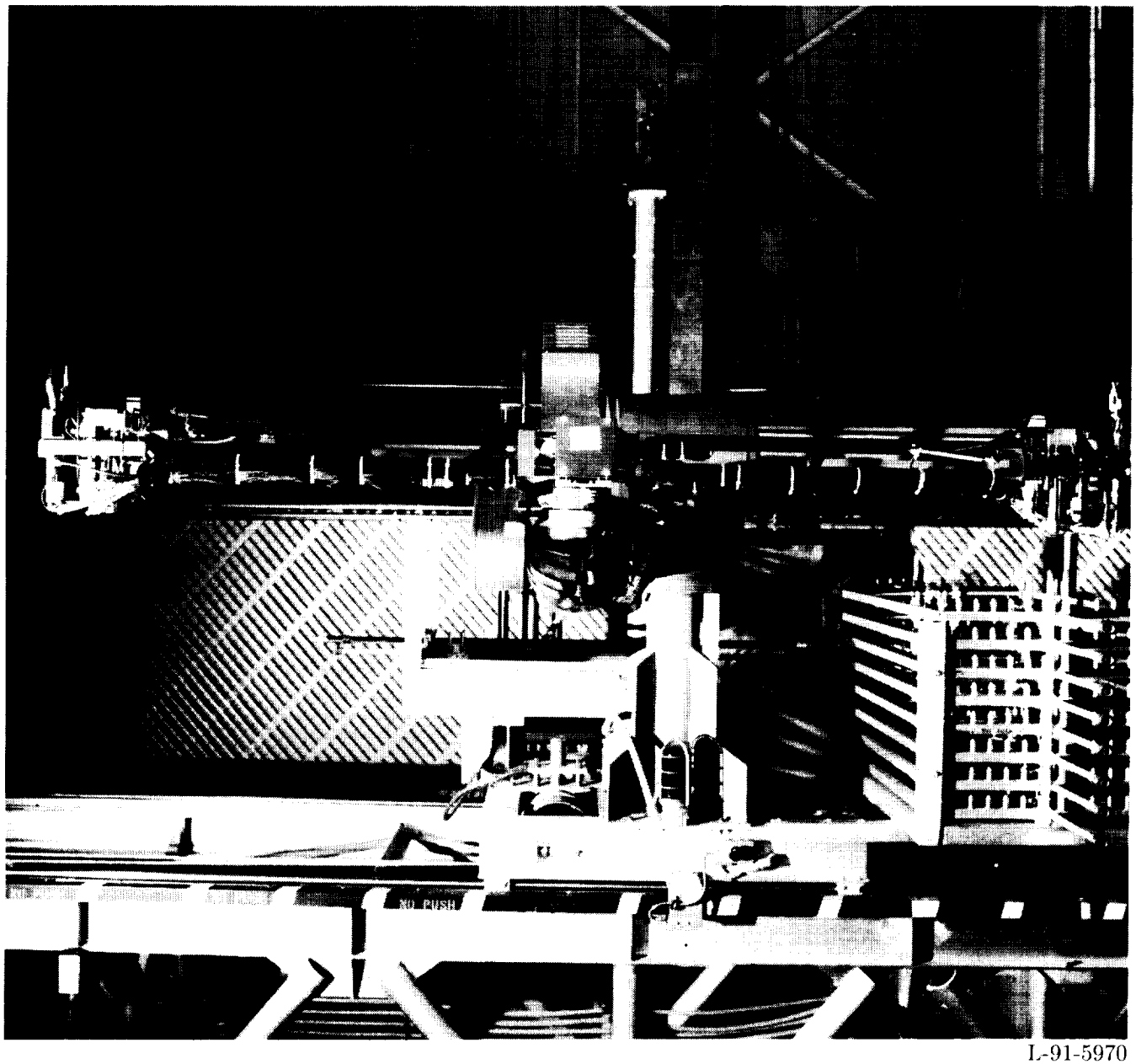
L-91-13 388

(b) Photograph.

Figure 1. Automated Structures Assembly Laboratory.

L-92-09366

Figure 2. Tetrahedral truss with hexagonal panels.

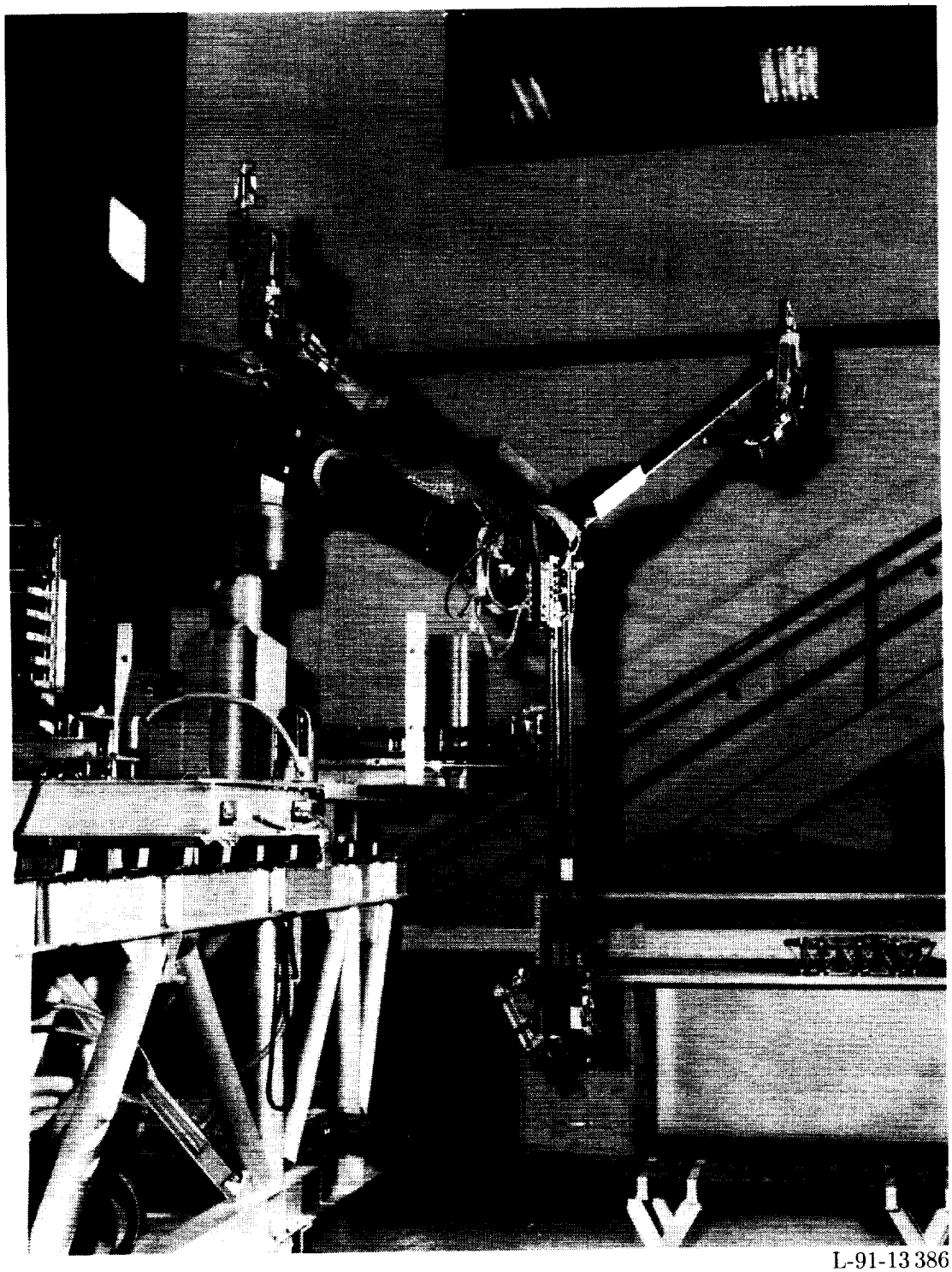


L-91-5970

(a) Strut end effector.

Figure 3. ASAL end effectors.

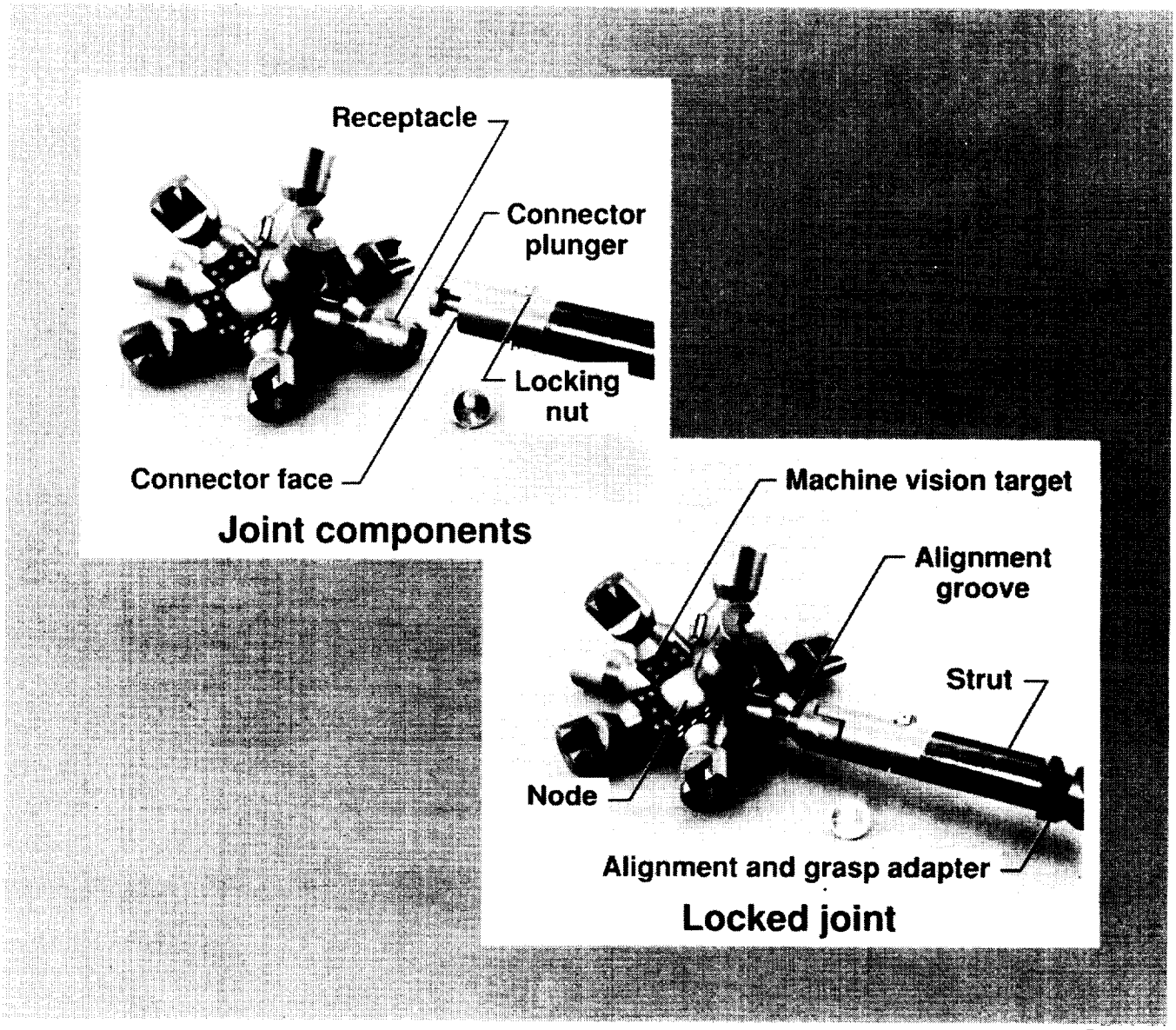
ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH



L-91-13 386

(b) Panel end effector.

Figure 3. Concluded.



L-93-24

Figure 4. Truss joint and node hardware.

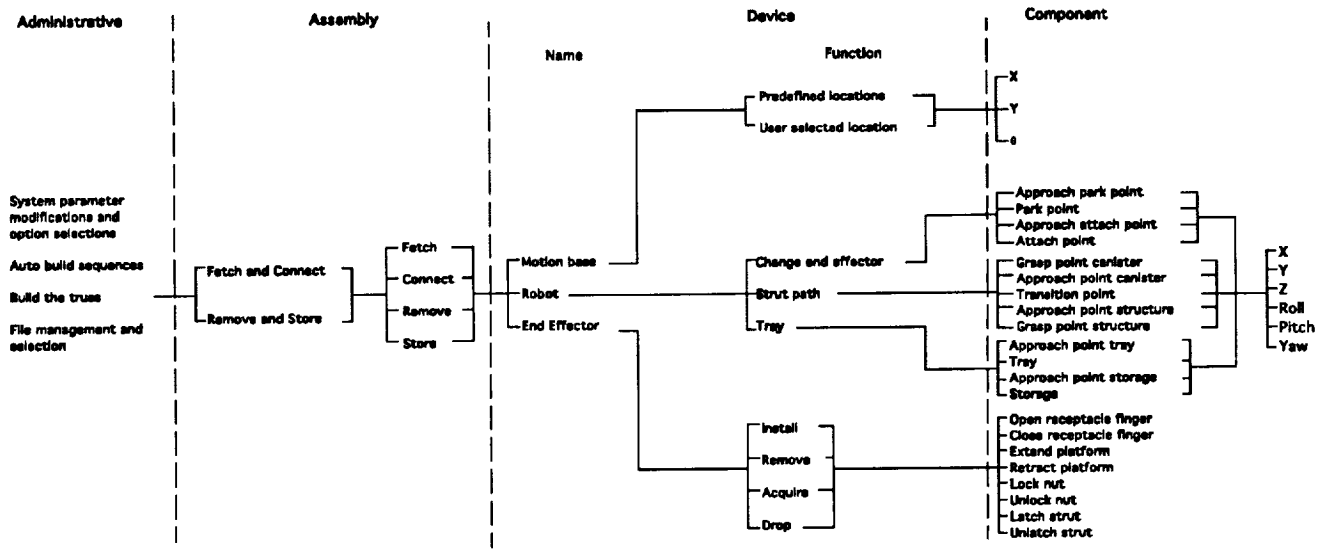


Figure 5. Software design layout.

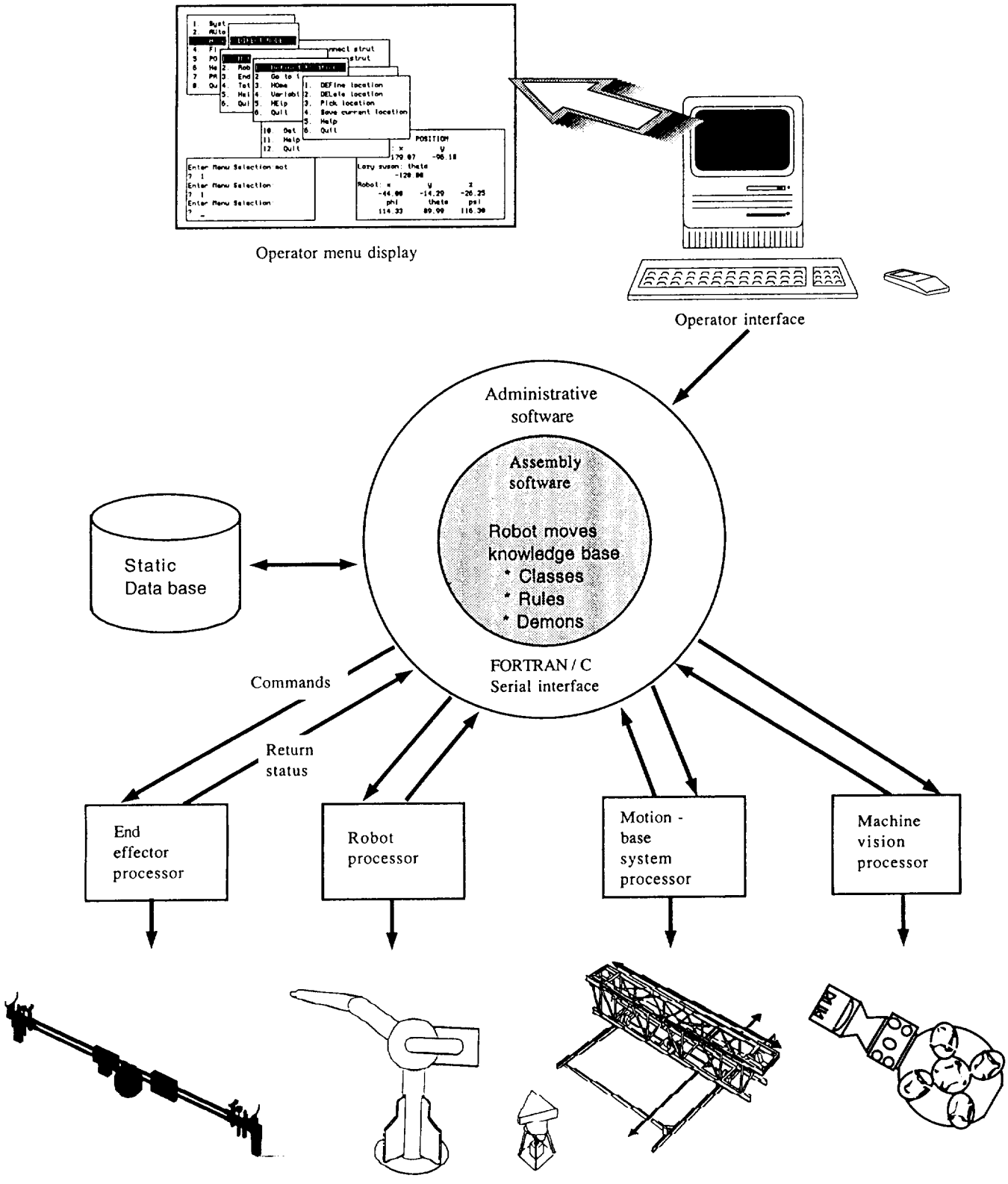


Figure 6. ASAL software architecture.

```
Classes:

 STRUTS:
      attributes:
            OBSERVER NAME: str.
            ALTERNATE NAME: str.
            ROBOT NAME: str.
            TRAY: int.
            SLOT: int.
            NODE END: str.
            CAP END: str.
            WHERE: sgl (CANISTER, INSTALLED, ARM).
            FLIP: str.
            CAN_FLIP: truth.
            NODE DIRECT: str.
            MB_INDEX1: int.
            MB_INDEX2: int.
      %
 endclass.
```

Figure 7. Class definition of struts.

```
State:
 if
      current_state = AP_CAN* and
      phase = out
 then
     next_state = GP_CAN**
 endif.


Phase:
 if
      current_state = AP_CAN and
      target_state = GP_CAN and
      current_strut>where = CANISTER | ARM
 then
     phase = out
 endif.

*     AP_CAN : Canister approach point
**    GP_CAN: Canister grasp point
```

Figure 8. Example rules for strut-path determination.

```
State GP_CAN:
  when
   next_state = GP_CAN
  then
    reassert rule_flag = false.
     erase status_mode.
(a)  reassert check_scar = true.
(b)  if ee response = reversed then
(c)     reassert return = true.
    else                  \ ee response = worked
        if ((init = false and restart = false) or override) and
          status_mode = false then
(d)       message "COMMAND$reset fts".
      endif.
(e)     if current strut>CAN_FLIP then
(f)      reassert tomerl = "GOTO GP_FLIP_CAN*".
      else
(g)      reassert tomerl = "GOTO GP_CAN*".
      endif.
        if determined (current strut) then
(h)        reassert tomerl = combine(tomerl,current strut>SLOT,"*",
                    current strut>TRAY).
      endif.
(i)    reassert send merlin = true.
       if halt_op = false then
          if robot success then
(j)         message "UPDATE$charstate,GP_CAN".
             reassert current_state = GP_CAN.
          else    \ return to calling state
            if current strut>CAN_FLIP then
(k)            reassert tomerl = combine("GOTO REV_GP_FLIP",
                          current strut>SLOT,"*",current
                        strut>TRAY).
               reassert send merlin = true.
           endif.
(l)          reassert return = true.
         endif.
       endif.
    endif.
    endwhen.
```

Figure 9. Demon for moving robot to canister grasp point.

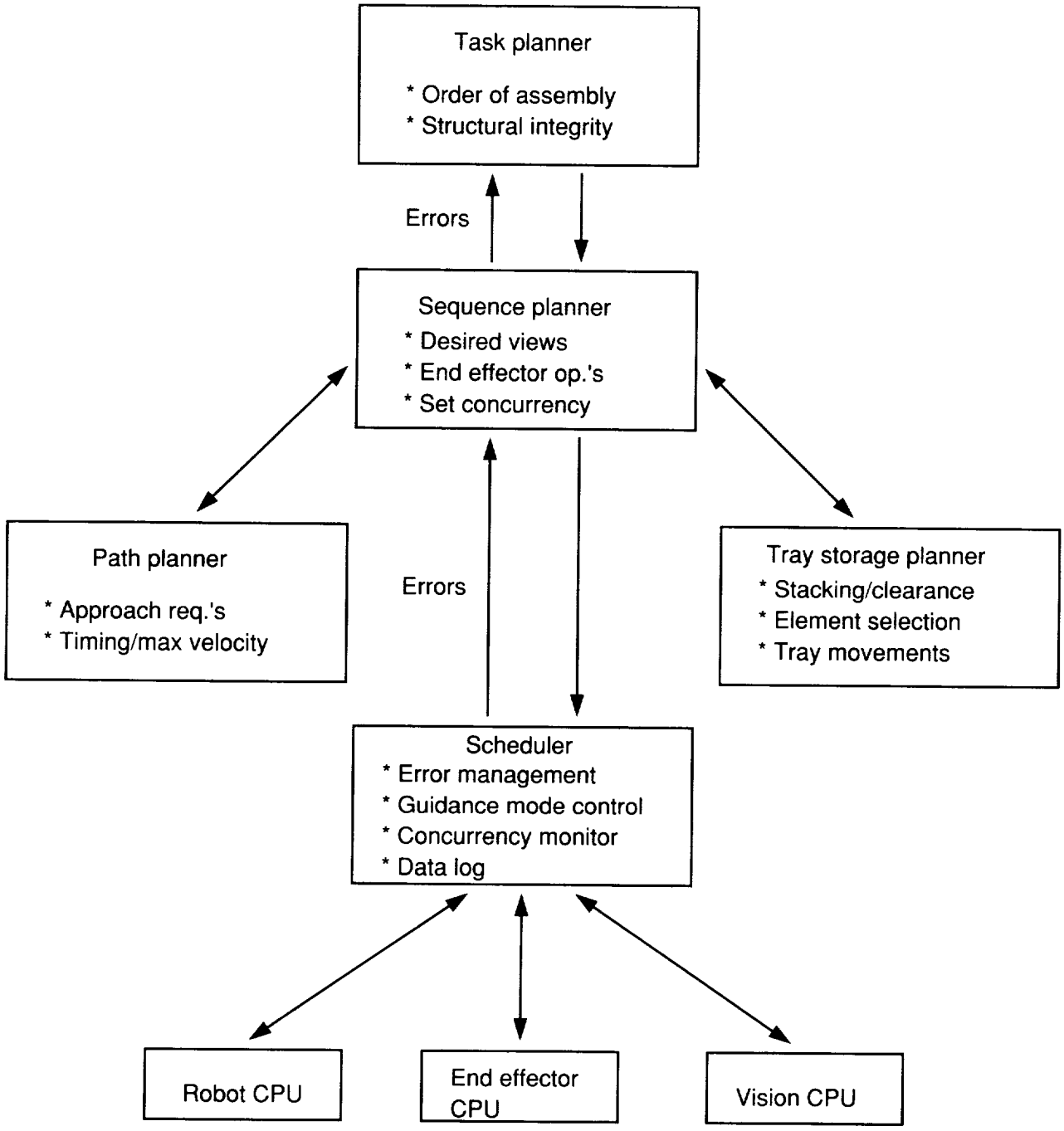


Figure 10. ASAL-distributed architecture.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November 1993	Technical Memorandum

4. TITLE AND SUBTITLE
An Expert System Executive for Automated Assembly of Large Space Truss Structures

5. FUNDING NUMBERS
WU 586-02-11

6. AUTHOR(S)
Cheryl L. Allen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23681-0001

8. PERFORMING ORGANIZATION REPORT NUMBER
L-17189

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA TM-4487

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited

Subject Category 63

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Langley Research Center developed a unique test bed for investigating the practical problems associated with the assembly of large space truss structures using robotic manipulators. The test bed is the result of an interdisciplinary effort that encompasses the full spectrum of assembly problems - from the design of mechanisms to the development of software. This paper describes the automated structures assembly test bed and its operation, details the expert system executive and its development, and discusses the planned system evolution. Emphasis is on the expert system implementation of the program executive. The executive program must direct and reliably perform complex assembly tasks with the flexibility to recover from realistic system errors. The employment of an expert system permits information that pertains to the operation of the system to be encapsulated concisely within a knowledge base. This consolidation substantially reduced code, increased flexibility, eased software upgrades, and realized a savings in software maintenance costs.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Expert system; Automated Structures Assembly Laboratory (ASAL); Robot; Truss structure	19
	16. PRICE CODE: A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		

NSN 7540-01-280-5500

Standard Form 298(Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102